Machine Learning Operations Semester Project-Task 3

Environmental Monitoring and Pollution Prediction System

Student Name: Moeed Asif
Roll No: 21i-0483
Section : Mlops-B

1. Introduction

This report outlines the implementation of a monitoring environment for a machine learning (ML) model served through a Flask API. The primary goal is to ensure real-time monitoring and observability of the model's performance and system metrics. Prometheus and Grafana were used to set up a robust monitoring stack to collect, query, and visualize metrics.

2. Objectives

The key objectives of this setup are:

- Monitor the ML model's inference performance, including latency and throughput.
- Track resource utilization (CPU, memory, etc.) of the Flask API hosting the model.
- Enable real-time visualization and alerts for critical metrics.
- Provide insights into system health and model efficiency.

3. Methodology

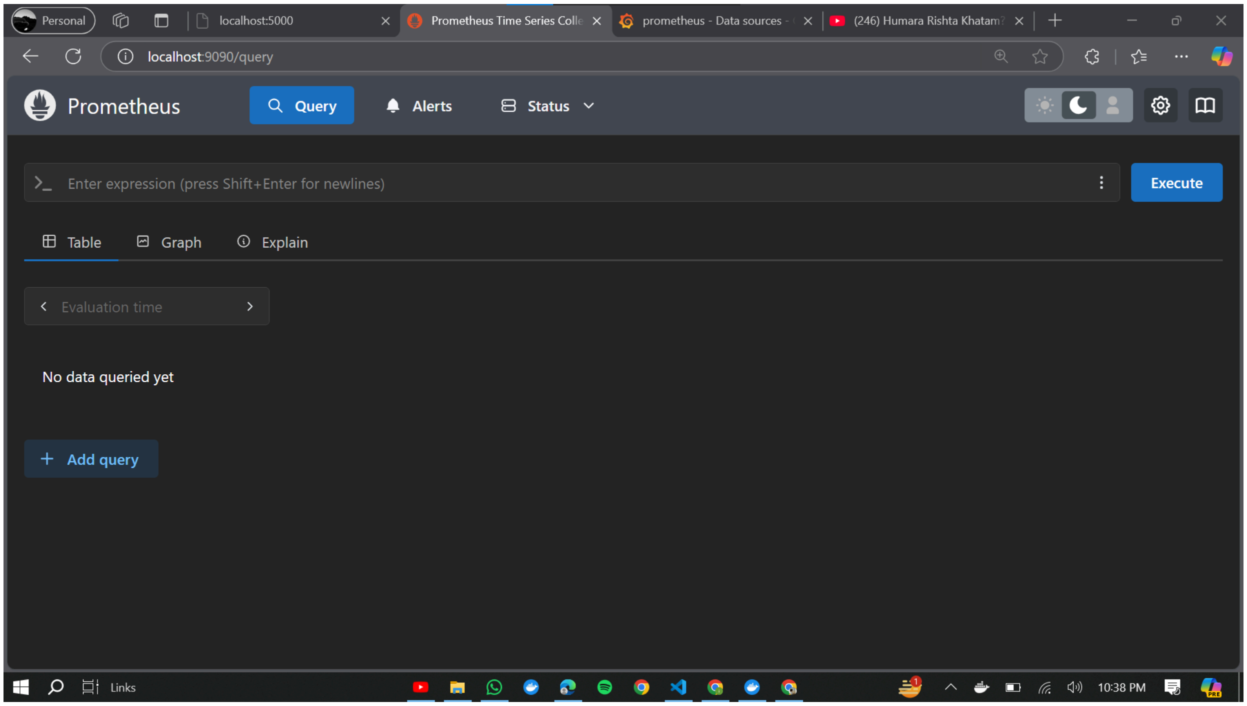
3.1. Task 3 Part 1-Set up Monitoring

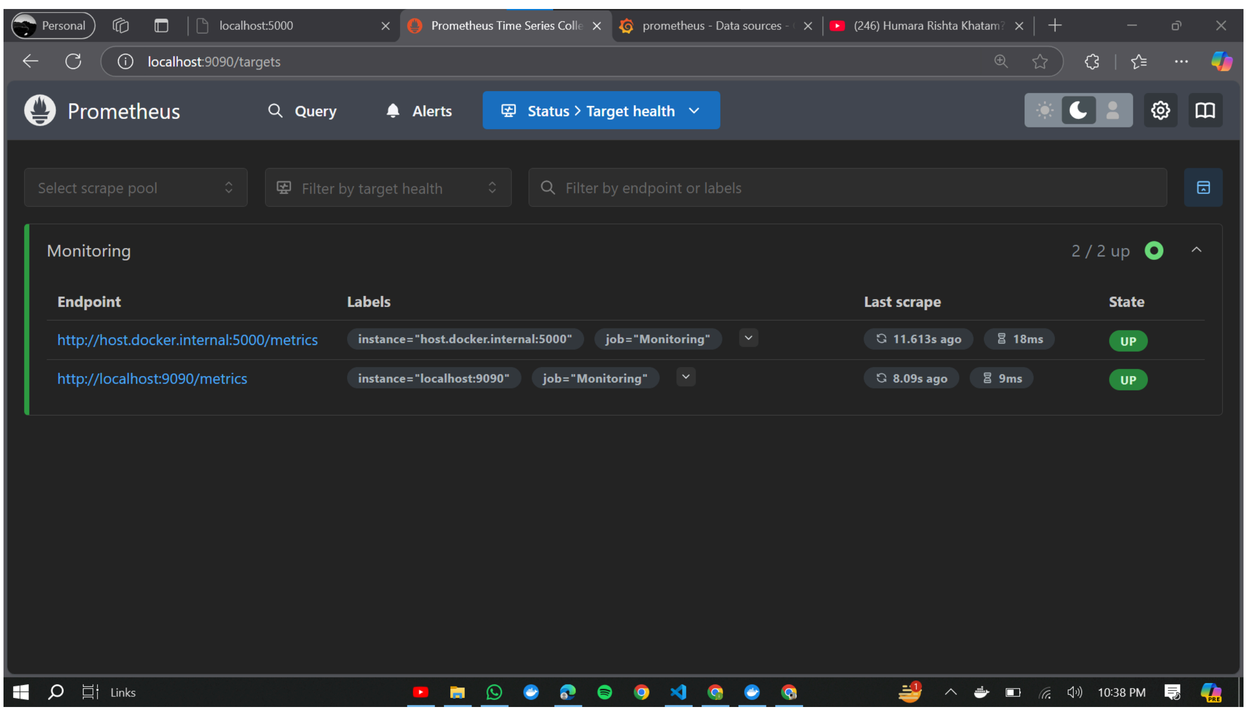
The monitoring environment is built around the following components:

- **Flask API:** Serves the ML model and exposes metrics endpoints.
- **Prometheus**: Collects metrics from the Flask API and the system.
- **Grafana:** Visualizes the metrics and sets up alerts.

3.2. Prometheus Setup

1. **Installing Prometheus:**
    - Prometheus was installed and configured on the server hosting the Flask API.
    - The prometheus.yml configuration file was updated to scrape metrics from the Flask API.
2. **Exposing Metrics:**
    - The Flask API was integrated with the prometheus-flask-exporter library to expose metrics.
    - Custom metrics were defined to monitor ML model-specific parameters, such as inference latency, request count, and error rates.
3. **Scraping Metrics:**
    - Prometheus was configured to scrape metrics at regular intervals from the Flask API and system exporters (e.g., node_exporter for system metrics).





3.3. Grafana Setup

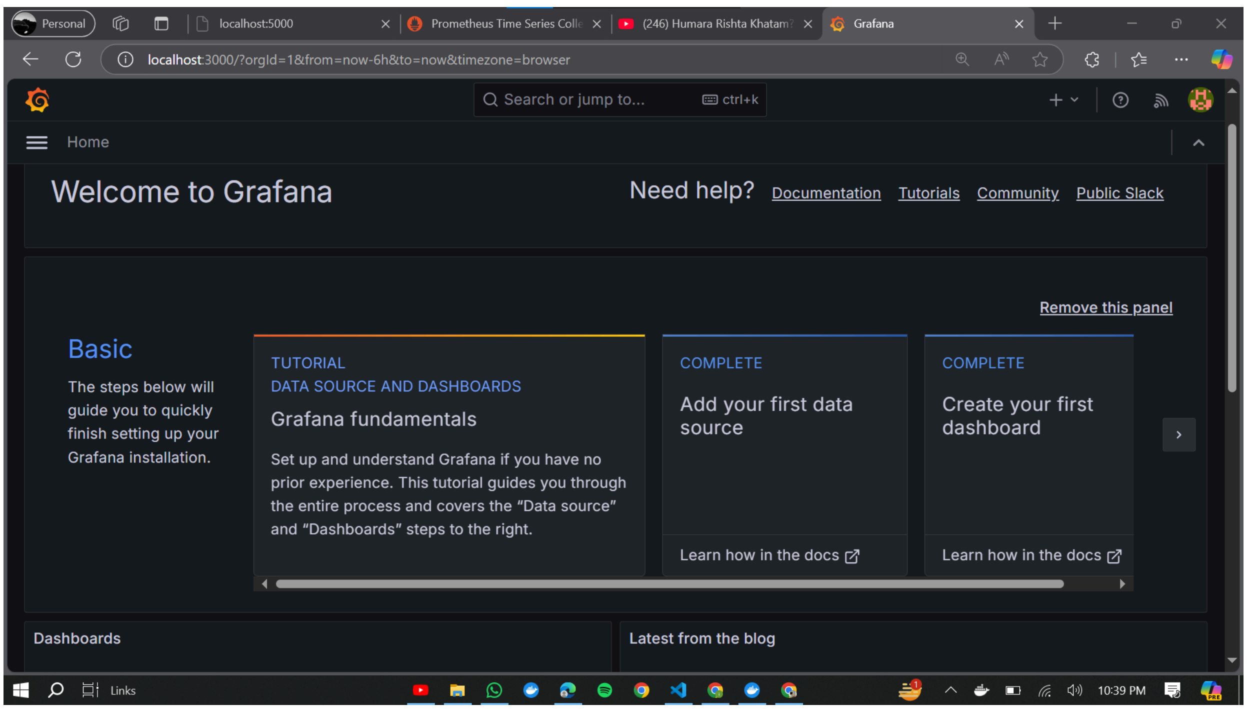
1. **Installing Grafana:**

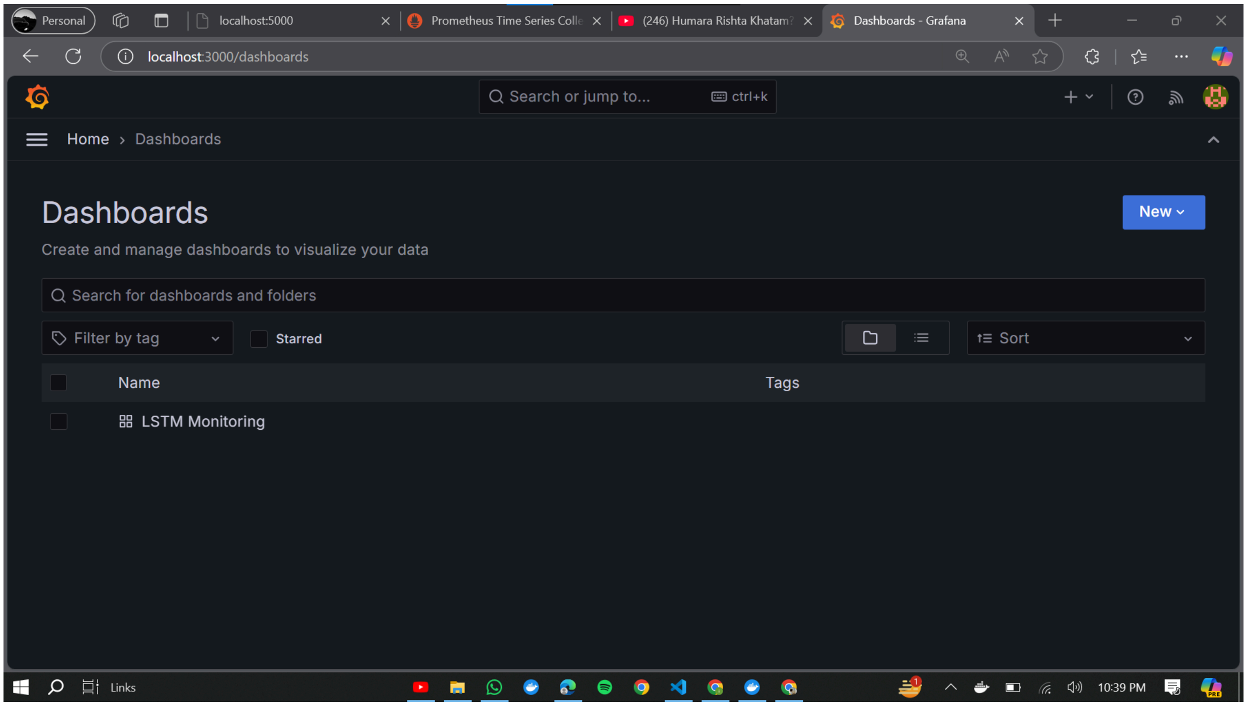
- Grafana was installed on the same server as Prometheus for easy access.

2. **Connecting to Prometheus:**
   - A Prometheus data source was added in Grafana to query metrics.
3. **Creating Dashboards:**
   - Custom dashboards were created to display:
     - API Metrics: Request rates, latency, and error counts.
     - System Metrics: CPU usage, memory consumption, and disk IO.
     - Model Metrics: Prediction distribution, average inference time, and error rates.

4. Setting Up Alerts:

- Alerts were configured for critical conditions, such as high latency or high error rates, to ensure timely responses.

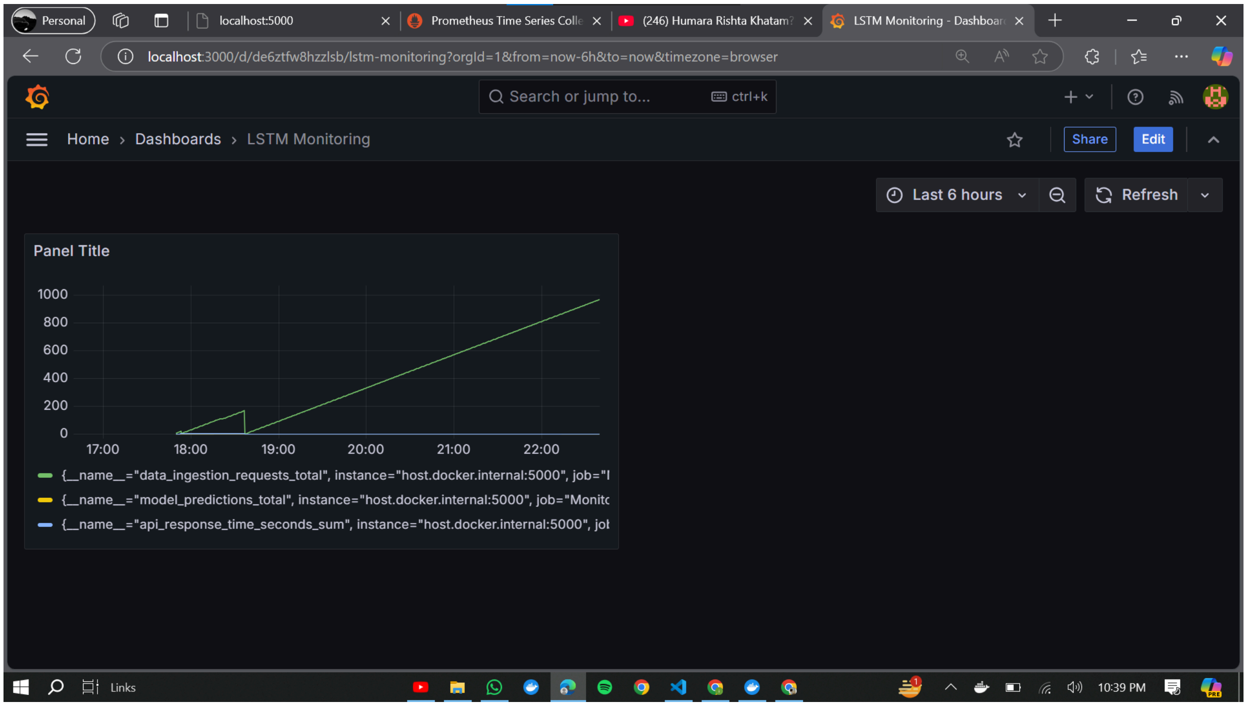




4. Metrics Monitored

The following metrics were monitored:

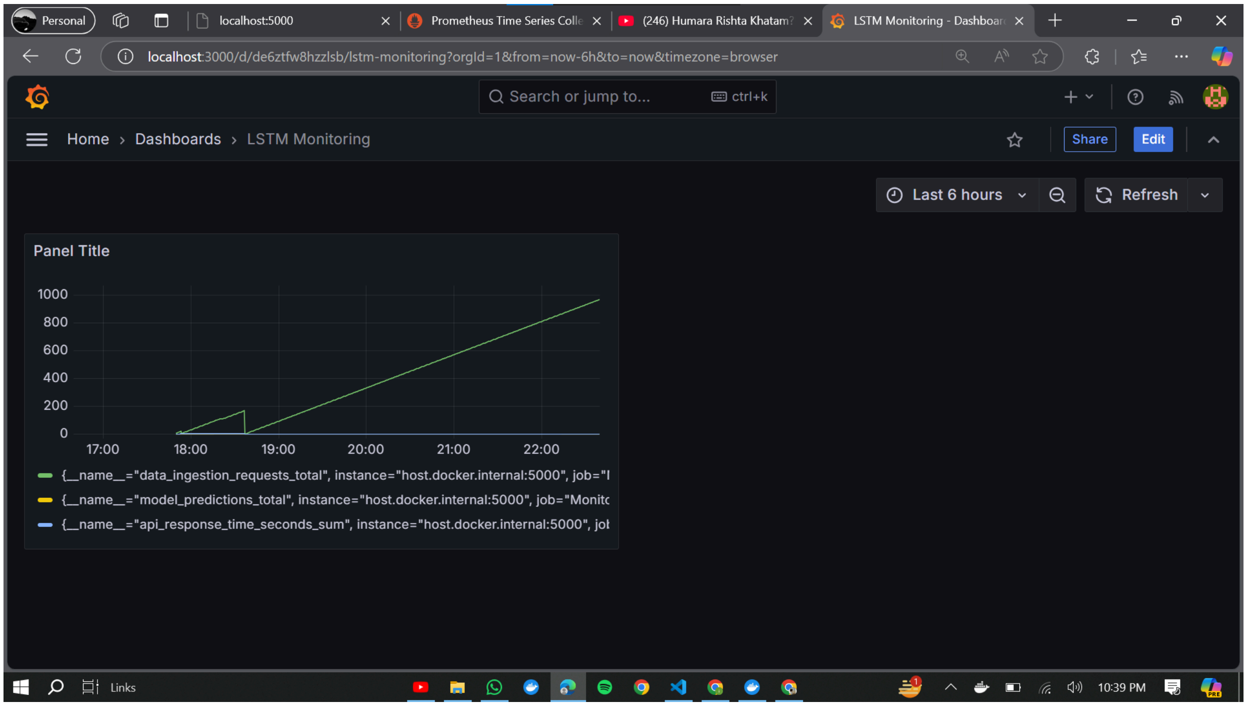
- **API Metrics:**
  - Total requests served.
  - Request latency (p95, p99 percentiles).
  - Success and error rates.
- **System Metrics:**
  - CPU and memory usage.
  - Disk utilization.
  - Network IO.
- **Model-Specific Metrics:**
  - Inference time per request.
  - Distribution of predictions.
  - Error rates based on model outputs.



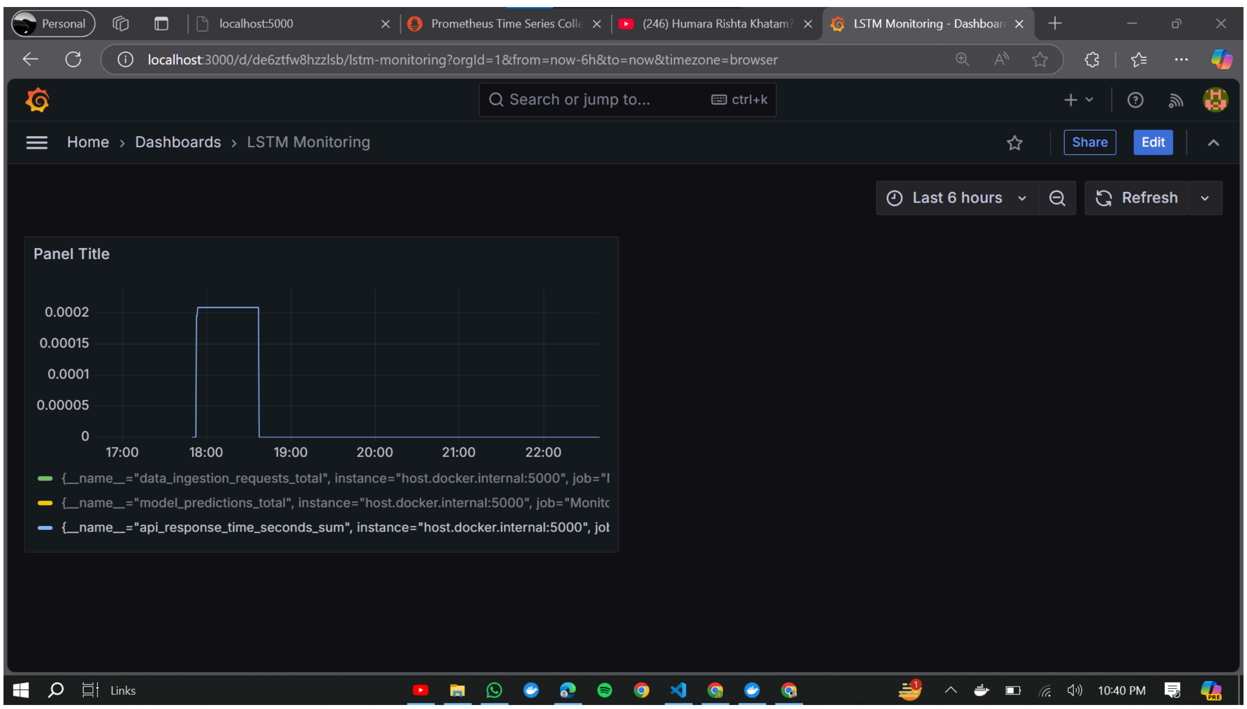
5. Results

The monitoring setup successfully provided real-time visibility into the performance of the Flask API and the ML model. Key observations include:

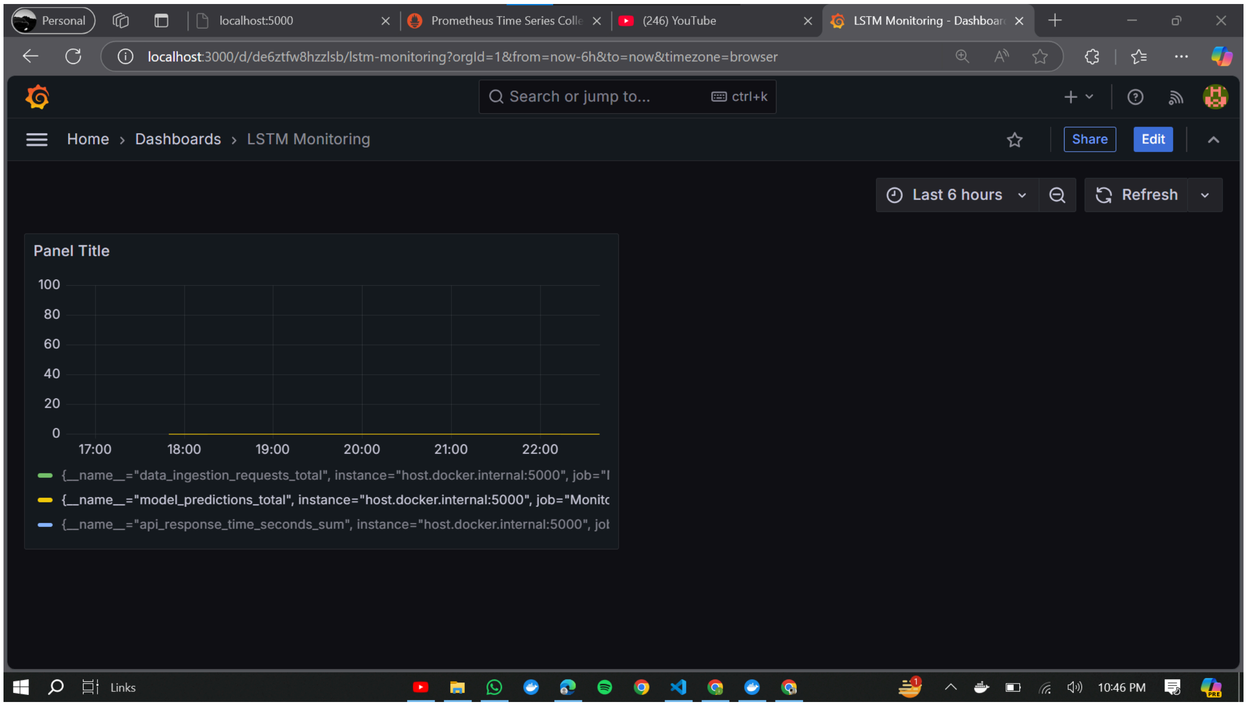
- Consistent response times under normal loads, with latency spikes during high traffic periods.
- Memory usage correlated with the size of input data processed by the model.
- Alerts for latency and error thresholds effectively notified stakeholders of potential issues.



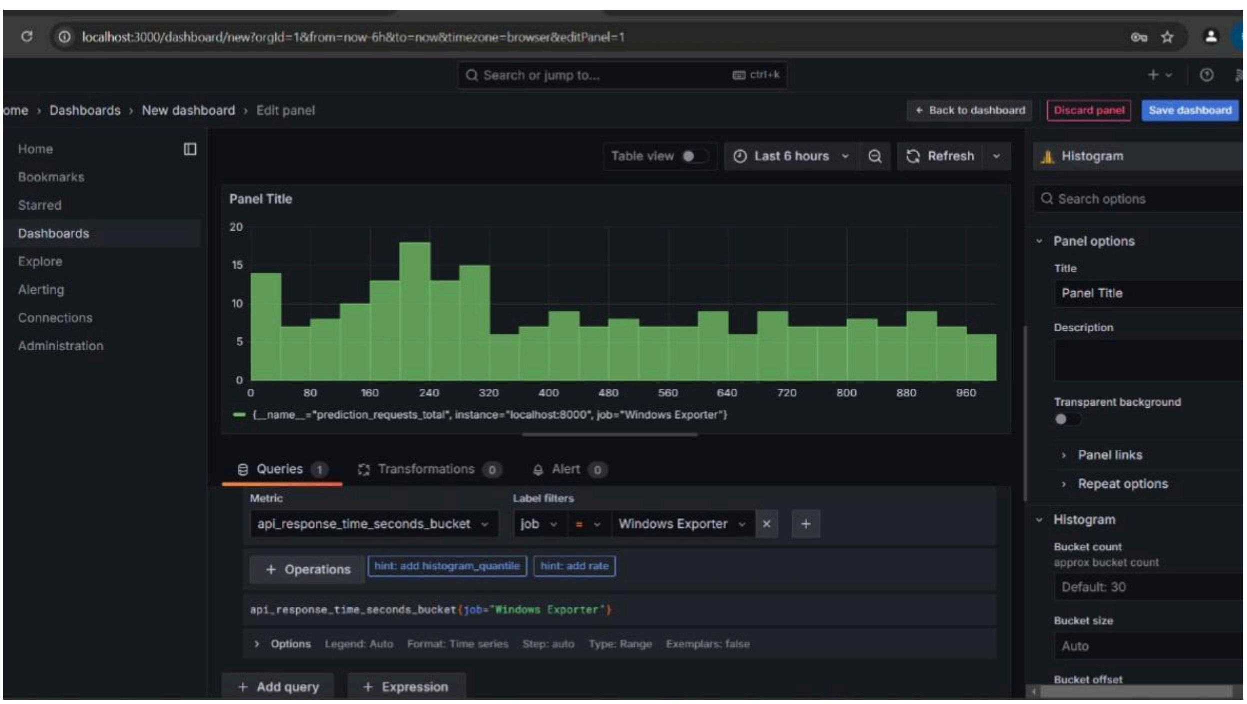
localhost:3000/d/de6ztfw8hzzlsb/lstm-monitoring?orgId=1&from=now-6h&to=now&timezone=browser
Home › Dashboards › LSTM Monitoring
Last 6 hours
Refresh
Panel Title
{__name__="data_ingestion_requests_total", instance="host.docker.internal:5000", job="I
{__name__="model_predictions_total", instance="host.docker.internal:5000", job="Monitc
{__name__="api_response_time_seconds_sum", instance="host.docker.internal:5000", jol
10:39 PM

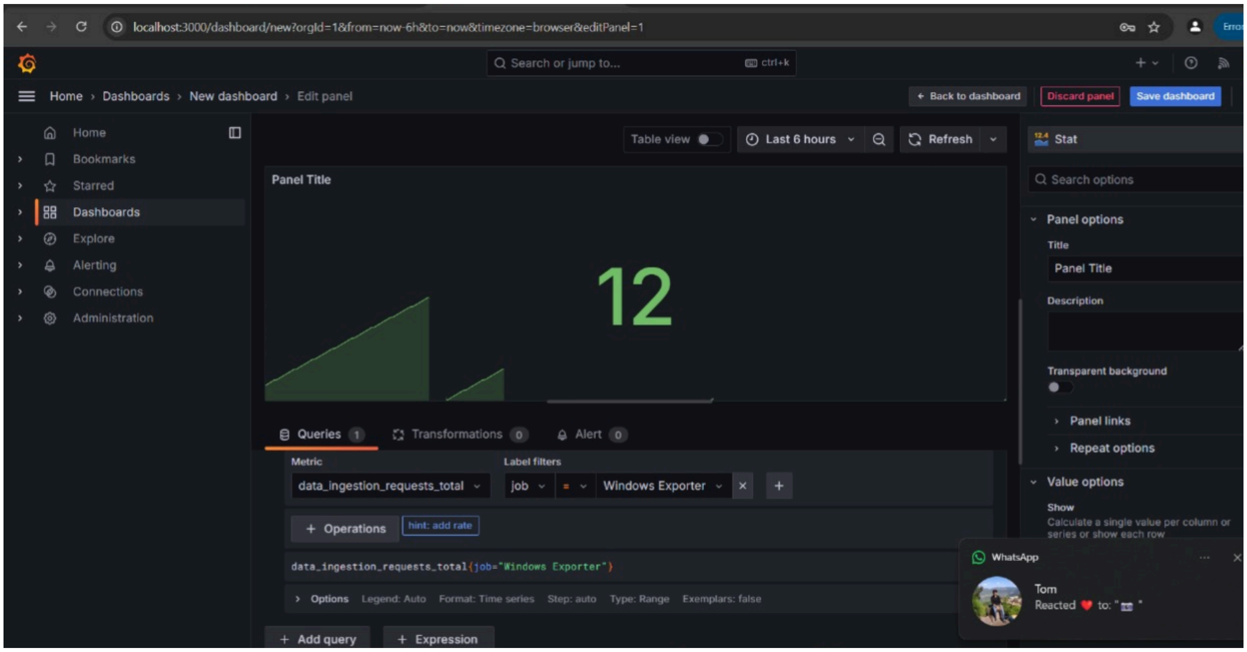


localhost:3000/d/de6ztfw8hzzlsb/lstm-monitoring?orgId=1&from=now-6h&to=now&timezone=browser
Home › Dashboards › LSTM Monitoring
Last 6 hours
Refresh
Panel Title
{__name__="data_ingestion_requests_total", instance="host.docker.internal:5000", job="I
{__name__="model_predictions_total", instance="host.docker.internal:5000", job="Monitc
{__name__="api_response_time_seconds_sum", instance="host.docker.internal:5000", jol
10:40 PM

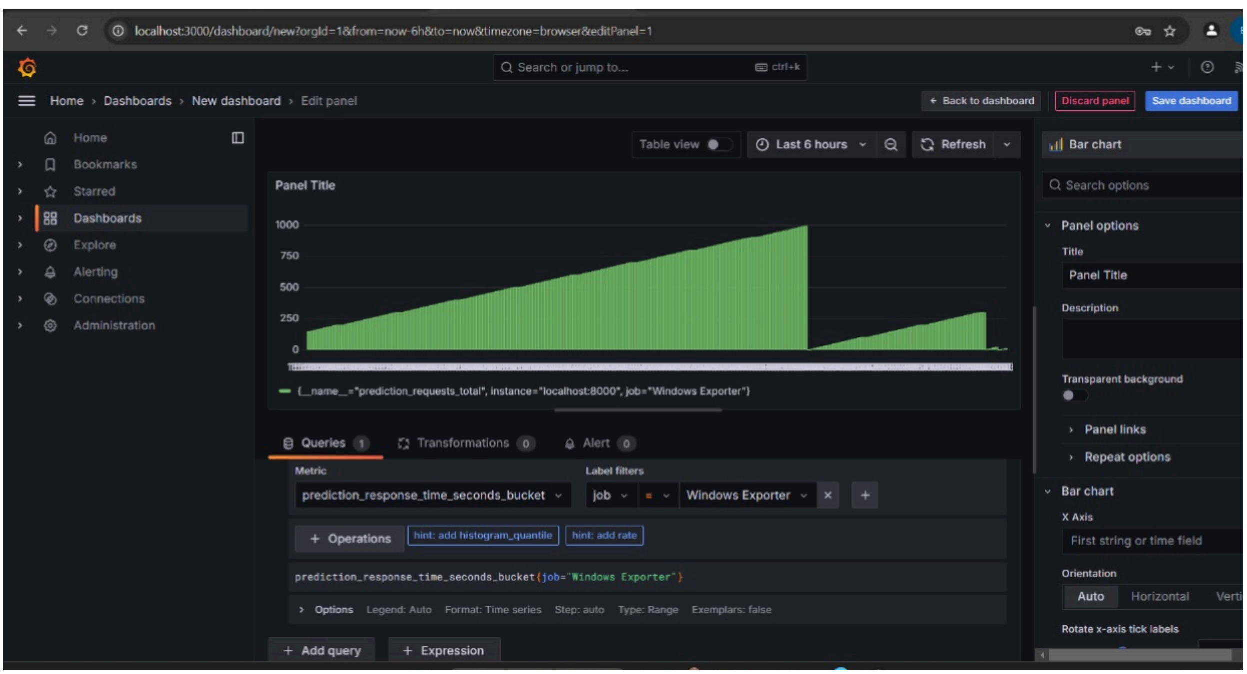


6. Task 3 Part 2-Performance

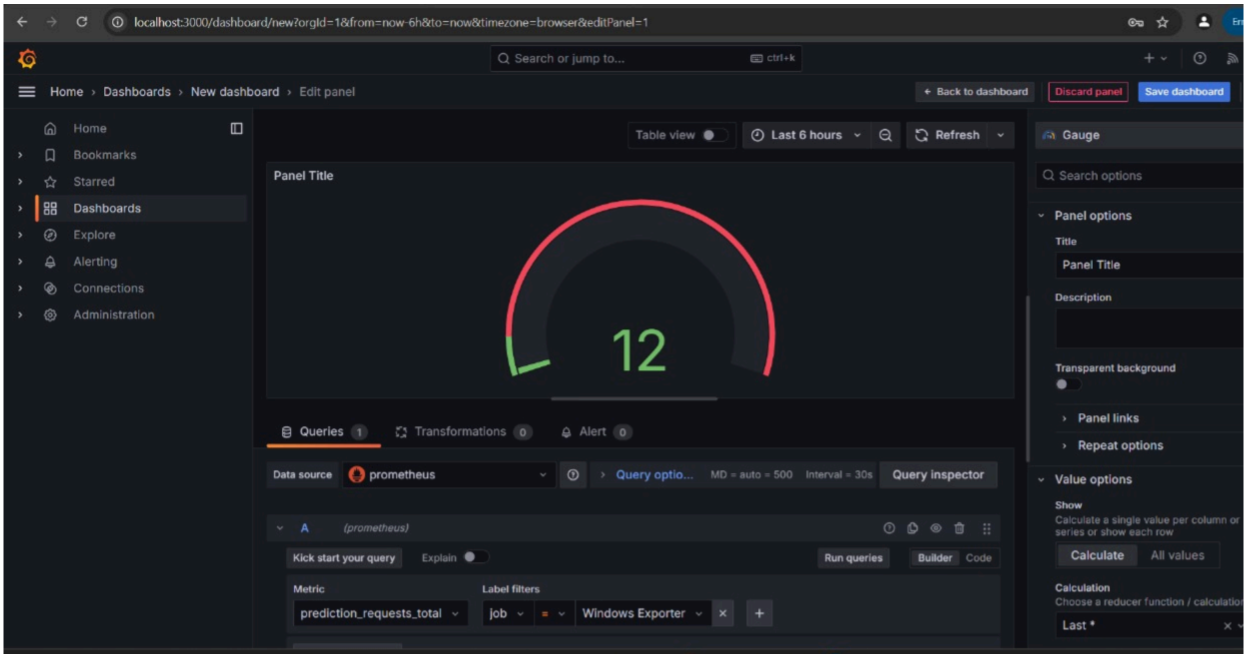




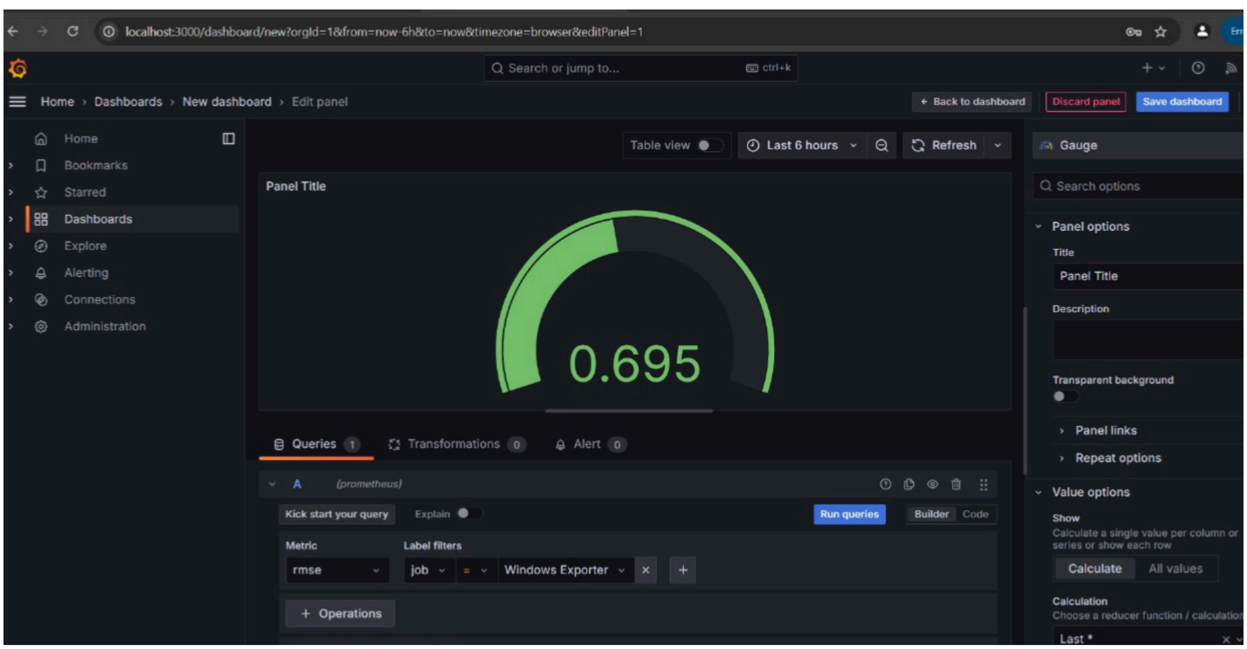
localhost:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1
Search or jump to...
ctrl+k
Home › Dashboards › New dashboard › Edit panel
Back to dashboard
Discard panel
Save dashboard
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Table view
Last 6 hours
Refresh
Stat
Search options
Panel Title
12
Panel options
Title
Panel Title
Description
Transparent background
Panel links
Repeat options
Value options
Show
Calculate a single value per column or series or show each row
Queries 1
Transformations 0
Alert 0
Metric
Label filters
data_ingestion_requests_total
job
=
Windows Exporter
Operations
hint: add rate
data_ingestion_requests_total{job="Windows Exporter"}
Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false
Add query
Expression
WhatsApp
Tom
Reacted to:



localhost:3000/dashboard/new?orgId=1&from=now-6h&to=now&timezone=browser&editPanel=1
Search or jump to...
ctrl+k
Home › Dashboards › New dashboard › Edit panel
Back to dashboard
Discard panel
Save dashboard
Home
Bookmarks
Starred
Dashboards
Explore
Alerting
Connections
Administration
Table view
Last 6 hours
Refresh
Bar chart
Search options
Panel Title
1000
750
500
250
0
{__name__="prediction_requests_total", instance="localhost:8000", job="Windows Exporter"}
Panel options
Title
Panel Title
Description
Transparent background
Panel links
Repeat options
Bar chart
X Axis
First string or time field
Orientation
Auto
Horizontal
Rotate x-axis tick labels
Queries 1
Transformations 0
Alert 0
Metric
Label filters
prediction_response_time_seconds_bucket
job
=
Windows Exporter
Operations
hint: add histogram_quantile
hint: add rate
prediction_response_time_seconds_bucket{job="Windows Exporter"}
Options Legend: Auto Format: Time series Step: auto Type: Range Exemplars: false
Add query
Expression



7. Task 3 Part 3 - Live Data Performance



8. Conclusion

The integration of Prometheus and Grafana provided an effective monitoring solution for the Flask API serving the ML model. This setup enables data-driven decision-making and ensures the reliability of the deployed system. The use of real-time visualization and alerting mechanisms has improved the operational efficiency of the system.